


PATENT
5053-28501

"EXPRESS MAIL" MAILING LABEL
NUMBER EL690353481US

DATE OF DEPOSIT SEPTEMBER 29, 2000

I HEREBY CERTIFY THAT THIS PAPER OR
FEE IS BEING DEPOSITED WITH THE
UNITED STATES POSTAL SERVICE
"EXPRESS MAIL POST OFFICE TO
ADDRESSEE" SERVICE UNDER 37 C.F.R. §
1.10 ON THE DATE INDICATED ABOVE
AND IS ADDRESSED TO THE
COMMISSIONER OF PATENTS AND
TRADEMARKS, WASHINGTON, D.C. 20231



Derrick Brown

"CONDITION COMPONENT FRAMEWORK FOR REINSURANCE"

By:

Bjørnar Evenshaug
Harald Sverdrup-Thygeson

PRIORITY CLAIM

This application claims the benefit of U.S. Provisional Application No. 60/157,223 entitled "CONDITION COMPONENT FRAMEWORK FOR REINSURANCE," filed September 30, 1999.

BACKGROUND OF THE INVENTION

1. Field of the Invention

10

The present invention generally relates to computer software. More particularly, the present invention relates to the customization of conditions of a business contract.

2. Description of the Related Art

15

Financial services organizations have come to rely on complex software systems to administer their businesses. Financial service organizations may include, for example, reinsurance organizations. Reinsurance may include the transfer by a first insurer of all or part of a risk to a second insurer to provide protection against the risk, as well as any associated transactions. In other words, reinsurance can be thought of as "insurance for insurance companies." When catastrophic events such as earthquakes, floods, tornadoes, hurricanes, airline accidents, etc. occur the insured often file for damage claims to reduce the impact of the loss of property and/or life. The insurance companies, which offer policies to cover for such catastrophic losses, come under severe financial strain to absorb the losses and still maintain the required surplus. The surplus is an insurance company's net worth, i.e. its assets minus its liabilities. Some insurance companies may deplete their surplus too far to be unable to fulfill their obligations to their policyholders. To protect policyholders against insolvency of an insurance company, government regulations may require insurance companies to maintain a minimum surplus. The size of an insurance

companies surplus is thus considered as an important factor to rate insurance companies. To maintain a surplus and protect against insolvency insurance companies may purchase their own insurance policies, i.e. reinsurance policies.

5 An insurance company, also known as primary or ceding company, may purchase a reinsurance policy from a reinsurance company, also known as reinsurer, in much the same way that direct or primary insurance is purchased. The primary or the first insurer may also be called a cedent, and the secondary or the second insurer may be called a reinsurer. Reinsurance organizations may include cedents, reinsurers, and any other
10 entities involved in reinsurance transactions. Reinsurance may protect a cedent against catastrophes and cumulative losses and also enable it to accept risks that exceed its own underwriting capacity.

 The complexity of the reinsurance field tends to require software for reinsurance
15 administration to be complex as well. Generally speaking, software for reinsurance administration may be expected to handle risk selection, portfolio analysis, policy administration, claims, accounting, and other areas vital to the reinsurance field. Reinsurance profits may depend on analysis of historical information, the ability to predict trends, and the ability to identify cumulative exposures within a current portfolio,
20 and reinsurance software may therefore be expected to meet requirements relating to those functions. Consequently, the development of a software system for reinsurance administration to meet the above-identified needs may require great time and expense.

 In a changing reinsurance market, a reinsurance company may quickly fall behind
25 its competitors. For example, as reinsurers begin taking on direct risks, the market may demand integrated solutions. Some reinsurance computer systems may not be able to easily and quickly migrate the graphical user interfaces from a text based display to a color, graphical display. Some reinsurance computer systems may be unable to modify and customize the condition component of a reinsurance contract, such as adjustment of

an insurance period, changing premium limits, etc. promptly in response to a customer requirement. A reinsurer may need to move quickly against competitors to meet the changing needs of the market or to revitalize a reinsurance product line. Therefore, it would be advantageous for a reinsurer to reduce the time, expense, and risk associated with a complex product development effort that may require more than one hundred person-years of development time. Even during development, technical and functional requirements may evolve as dozens of potential customers instruct the reinsurer of their customized needs. It would be further advantageous for a reinsurer to have a flexible development process that can meet evolving customer requirements in the middle of a development effort.

Utilization of object-oriented software design techniques is one approach towards reducing development time. Object-oriented software design may include the use of objects and classes. An object may include an encapsulation of data and methods for manipulating the data. A class may include a template for an object. An object is therefore an instance of a class, and an object is created by instantiating it from a class. Object-oriented techniques, when used properly, may permit the re-use of previously written program code such as classes. The use of development frameworks is a related approach towards reducing development time. As used herein, a “framework” may include a set of classes or templates that embodies an abstract design for solutions to a number of related problems. Frameworks may be employed with object-oriented techniques to include flexible, customizable condition component of a reinsurance business contract. Frameworks may also include a flexible object-oriented graphical user interface software to process reinsurance transactions.

Therefore, there is a need for a system and method for developing improved flexible and customizable condition component of a reinsurance business contract. It is also desirable to develop graphical user interface software for processing reinsurance business transactions with decreased development time and increased flexibility.

SUMMARY OF THE INVENTION

The present invention provides various embodiments of an improved method and system for developing improved flexible and customizable condition component of a reinsurance business contract. The method and system may be used by reinsurers to build an improved flexible, customized condition component associated with a reinsurance contract with increased re-use of existing functionality and decreased development time. A business process framework may include common functionality, e.g. a GUI for a plurality of business processes. The business process framework may include a template for executing the business processes. In one embodiment, the method may include deriving one or more business processes from the business process framework. One or more business process subclasses may be derived from one or more abstract business process classes, wherein the business process framework includes the one or more abstract business process classes. At run-time during the execution of the reinsurance administration system, the business process objects may be instantiated as instances of the one or more business process subclasses. The method may further include combining the business process subclasses to build one or more application programs, such as application programs for reinsurance transaction processing. In one embodiment, a first business process subclass may be configured to invoke a second business process subclass.

The business process framework may include a plurality of support processes. The method may further include identifying one or more of the support processes for each of the business process subclasses. The support processes may include, for example, support processes for data validation, support processes for performing syntax validation, support processes for security checks, support processes for graphical user interface elements, and support processes for persistent data storage, and/or support processes for performing precondition checks. In one embodiment, identifying one or

more of the support processes for each of the business process subclasses may include overriding one or more of hook methods for each of the business process subclasses.

5 The business process framework may include a plurality of hook methods. The hook methods may permit the developer to access points during the execution of the business process. By overriding the hook methods at various points, the developer may identify methods to be performed at various points during execution of the business process. Overriding one or more of the hook methods for each of the business processes may include replacing one or more of the hook methods inherited from an abstract
10 business process class with one or more new methods. The hook methods may include, for example, methods to be executed during initialization of the business process, methods to be executed prior to execution of a data entry support process, methods to be executed during execution of data entry without a user interface, methods to be executed prior to execution of a database commit support process, and/or methods to be executed
15 prior to execution of a database rollback support process. As will be apparent to one skilled in the art, many more hook methods may be employed in developing a business process.

20 The common functionality of the business process framework may include an order for executing steps in a business process. The steps in the business process may include pre-execution, data entry, data validation, pre-commit to a database, commit to the database, and post-commit to the database. The common functionality of the business process framework may include behavior for buttons in a graphical user interface, formatting and displaying windows in a graphical user interface, methods or variables of
25 the abstract business process classes which are inherited by the business process subclasses, logging and displaying error messages, committing changes to a database, and/or rolling back changes to a database.

5 An object-oriented, multi-dimensional reinsurance contract framework may represent a reinsurance contract. The multi-dimensional reinsurance contract framework may permit the handling of various dimensions of a reinsurance contract such as handling different versions of the reinsurance contract (e.g., yearly periods), handling multi-level specialization hierarchy (e.g., different conditions for different sections of the reinsurance contract, handling the change of conditions within a period, and processing inheritance of contract structure and conditions from an abstract or master reissue contract. The multi-dimensional reinsurance contract framework may capture the various dimensions and hide the multi-dimensionality from subsequent domain modeling and the design of additional reinsurance contract components. The framework may also encapsulate business rules for managing the various dimensions into an abstract framework.

15 An object-oriented condition component framework may represent the condition component or dimension of a reinsurance contract. The condition component framework may permit the addition and/or modification of one or more conditions included in the condition component of the reinsurance contract. These conditions may comprise, for example, configuring or changing the premium limits, consolidation conditions, and other conditions for a reinsurance contract. The condition component may be implemented as reusable objects.

20 The business process framework may also include an object-oriented graphical user interface software. The graphical user interface may include a plurality of displays, along with a navigational tool to enable users to navigate through a hierarchy of displays related to reinsurance transaction processing. Each display may be defined as an object. A display may include one or more windows and the navigational tool. The one or more windows and the navigational tool may be displayed in a tiled, non-overlapping manner. The navigational tool may include one or more tool panels. The navigational tool may also include one or more user interface items such as icons and buttons to enable the user in navigating through the hierarchy of displays. Each user interface item included in the

navigational tool may be configured to perform a particular action such as calling a particular display on receiving a user input. The user may process a reinsurance business transaction by accessing the hierarchy of displays with the navigational tool integrated with the business process framework.

5

Attorney's Office
Conley, Rose & Tayon, P.C.
1000 Peachtree Street, N.E.
Atlanta, Georgia 30309
Phone: (404) 525-8800
Fax: (404) 525-8801
Email: info@conley-rose.com

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is an illustration of a typical computer system which is suitable for implementing various embodiments;

5 Figure 2 is an illustration of a distributed computing environment which is suitable for implementing various embodiments;

Figure 3a is an illustration of reinsurance transaction processing software using object-oriented technology according to one embodiment;

10 Figure 3b is an illustration of a layered architecture of a reinsurance administration system according to one embodiment;

Figure 4 is an illustration of a development process for developing a reinsurance administration system according to one embodiment;

15 Figure 5 is a flowchart illustrating the development of a reinsurance administration system through a business process framework according to one embodiment;

Figures 6a and 6b are state diagrams illustrating the life of a business process according to one embodiment;

Figure 7a illustrates one embodiment of an inheritance view of a hierarchy of business objects;

20 Figure 7b illustrates one embodiment of an inheritance view of a section object; and

Figures 8a through 8d illustrate one embodiment of a graphical user interface for processing a reinsurance business transaction.

25 While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and

DETAILED DESCRIPTION OF THE DRAWINGS

Figure 1: A Typical Computer System

5 Figure 1 illustrates a typical computer system 150, which is suitable for implementing various embodiments. Each computer system 150 typically includes components such as a CPU 152 with an associated memory medium such as floppy disks 160. The memory medium may store program instructions for computer programs, wherein the program instructions are executable by the CPU 152. The computer system
10 150 may further include a display device such as a monitor 154, an alphanumeric input device such as a keyboard 156, and a directional input device such as a mouse 158. The computer system 150 is operable to execute the computer programs to implement the system and method for developing a reinsurance transaction processing system, also referred to as the reinsurance administration system, as described herein. As described
15 herein, one embodiment of the typical the computer system 150 may be a reinsurance transaction processing system 150.

 The computer system 150 preferably includes a memory medium on which computer programs according to various embodiments may be stored. The term “memory
20 medium” may include an installation medium, e.g., a CD-ROM, or floppy disks 160, a computer system memory such as DRAM, SRAM, EDO RAM, Rambus RAM, etc., or a non-volatile memory such as a magnetic media, e.g., a hard drive, or optical storage. The memory medium may include other types of memory as well, or combinations thereof. In addition, the memory medium may be located in a first computer in which the programs are
25 executed, or may be located in a second different computer, which connects to the first computer over a network (not shown). In the latter instance, the second computer provides the program instructions to the first computer for execution. Also, the computer system 150 may take various forms, including a personal computer system, mainframe computer system, workstation, network appliance, Internet appliance, personal digital assistant (PDA),

television system or other device. In general, the term "computer system" 150 can be broadly defined to encompass any device having a processor 152, which executes instructions from a memory medium.

5 The memory medium preferably stores software programs for reinsurance administration and/or software programs for developing a reinsurance administration system as described herein. The software program(s) may be implemented in any of various ways, including procedure-based techniques, component-based techniques, and/or object-oriented techniques, among others. For example, the software program may be
10 implemented using ActiveX controls, C++ objects, JavaBeans, Microsoft Foundation Classes (MFC), or other technologies or methodologies, as desired. A CPU, such as the host CPU 152, executing code and data from the memory medium includes a means for creating and executing the software program or programs according to the methods and/or block diagrams described below.

15 In Figure 1, an embodiment of a reinsurance transaction processing system may include a computer system 150. The term "computer system" as used herein generally includes the hardware and software components that in combination allow the execution of computer programs. The computer programs may be implemented in software,
20 hardware, or a combination of software and hardware. A computer system's hardware generally includes a processor (CPU 152), memory media, and Input/Output (I/O) devices. As used herein, the term "processor" generally describes the logic circuitry that responds to and processes the basic instructions that operate a computer system.

25 A computer system's 150 software generally includes at least one operating system such as Windows NT, which is available from the Microsoft Corporation. The operating system is a specialized software program that manages and provides services to other software programs on the computer system. Software may also include one or more programs to perform various tasks on the computer system and various forms of data to be

used by the operating system or other programs on the computer system. The data may include but are not limited to databases, text files, and graphics files. A computer system's software generally is stored in non-volatile memory, such as floppy disks 160, or on an installation medium. A program may be copied into a volatile memory when running on the
5 computer system 150. Data may be read into volatile memory as required by a program.

A server may be defined as a computer program that, when executed, provides services to other computer programs executing in the same or other computer systems. The computer system on which a server program is executing may also be referred to as a
10 server, though it may contain a number of server and client programs. In the client/server model, a server is a program that awaits and fulfills requests from client programs in the same or other computer systems. A server that may be used to process reinsurance transactions is the SICS system. The SICS system is supplied by Computer Sciences Corporation.

15

As used herein, the term "channel" generally describes any pathway that allows a user to interface with a computer system. Channels may include, but are not limited to, main office entry systems, branch office entry systems, kiosks, call centers, Internet systems, and electronic data interchange (EDI) systems. A channel may be a computer system. For
20 example, the computer system 150 shown in Figure 1 may be a channel used for a reinsurance office data entry system, a branch office entry system, a client computer system with a web browser or a component of the system used in a call center channel.

Each type of channel may require different implementation systems. For example, a
25 kiosk entry system may use a touch-oriented visual screen. Special programming implementations may be needed to transfer data from the kiosk to the computer system. A channel may be implemented so that the channel may only be used for selected types of business transactions. A kiosk system may be set up to only process certain business

transactions, such as providing pricing quotes for specified reinsurance coverage situations. An office data entry system may use a monitor 154, keyboard 156 and pointing device 158 (mouse) to allow for data entry into a computer system 150. A call center typically involves the oral communication of data to a person who enters the data into a computer system through a keyboard 156 and/or pointing device 158. Electronic data entry systems may transfer data from one computer system to another.

The reinsurance transaction processing system 150 may further include a display screen 154 connected to the computer system 150 and a reinsurance database (not shown) residing on an internal or external storage. The database may also be referred to as a repository. As used herein, a “database” may include a collection of information from which a computer program may select a desired piece of data. As used herein, a “reinsurance database” is used as a synonym for a “database” when included in or coupled to a reinsurance transaction processing system 150. Reinsurance transaction processing program, also referred to as application program software, may be stored in memory. As used herein, a “reinsurance transaction processing program”, also referred to as a reinsurance program or reinsurance software, may include a software program which is configured to execute reinsurance transactions, such as by estimating the value of the reinsurance premium, for example.

The reinsurance transaction processing system 150 may be used by a Reinsurance Company (RC) for various embodiments of a system and method for processing reinsurance transactions. As used herein, a reinsurance Company (RC) includes a business organization that provides reinsurance products and/or services to customers. More particularly, the reinsurance products may pertain to estimating premiums, damages, etc. For example, estimating a reinsurance premium for a reinsurance policy, providing an estimate for a reinsurance claim filed for property and casualty damage.

In one embodiment, on incurring property damage to its policyholders, a primary insurance company may file a reinsurance claim (RC) with their reinsurance organization to cover loss of property. A RC may utilize a computer-based reinsurance claim processing system to process reinsurance claim. In one embodiment, the transaction processing may include estimating a value associated with the filed reinsurance claim.

As used herein, an RC business transaction may be defined as a service of an RC. Examples of business transactions include, but are not limited to, reinsurance transactions such as: filing of reinsurance claims, application for additional reinsurance coverage, payment of premiums, application for new reinsurance coverage, and customized benefits, etc. For a reinsurance provider, examples of business transactions may include, but are not limited to, adding or deleting primary insurance companies, tracking payables and receivables, establishing new policies/contracts, renewing policies/contracts, changing policies/contracts, canceling policies/contracts, providing pricing quotes for policies, and processing claims based on policies. Business transactions may also include services related to customers, reinsurance providers, employers, reinsurance agents, investigators, regulatory agencies, etc.

As used herein, an RC reinsurance transaction processing includes a series of instructions executed by a computer system 150 for processing an RC's business transactions. An RC transaction processing system 150 may include one or more processing tasks. A processing task may include a sequence of one or more processing steps or an ordered list or a structured list of one or more processing steps, associated with the business transaction to be processed by the RC transaction processing system. In one embodiment, the sequence of steps may be fixed. In another embodiment the sequence of steps may be established dynamically, in real-time. In one embodiment, the sequence of one or more steps may include an initial step, a final step, one or more intermediary steps, etc. In one embodiment, an RC user may select steps to process a reinsurance claim in a sequential manner. In another embodiment, the RC user may select

steps to process a reinsurance claim in a random or arbitrary manner. Examples of processing steps may include, but are not limited to: receiving an input from a user of the RC reinsurance transaction processing system, reading a value from a database, updating a field in a database, displaying the results of a business transaction on a computer screen 154, etc.

In one embodiment, the reinsurance transaction processing system utilizes object-oriented technology to process reinsurance transaction. In another embodiment, processing of reinsurance transaction may utilize traditional programming languages and databases to achieve the same result. Reinsurance objects may be defined to represent or model real-world business features of reinsurance products and services. Examples of reinsurance objects may include, but are not limited to, objects representing the following: a reinsurance claim; an accident report; a settlement; an estimated claim; RC service facilities, customers, and employees; business process such as a new reinsurance application and calculation of a premium; interfaces to external reinsurance organizations; work tasks such as calculations, decisions, and assignments; temporal objects such as calendars, schedulers, and timers; and elemental data necessary to accomplish work tasks such as medical costs, risk factors, automobile repair costs, etc.

A reinsurance object may be represented on the computer screen 154 by a graphical icon or by a display listing the properties of the reinsurance object in graphic and/or alphanumeric format. In one embodiment, a reinsurance claim object may be configured to gather and evaluate data for processing a filed reinsurance claim and to automatically make decisions about the reinsurance claim. The one or more processing steps associated with the processing of a reinsurance claim may also be configured as one or more processing step objects. In one embodiment, a display screen 154, which also may be referred to as a page or a display, may be associated with a processing step. The display screen may also be represented as an object. Each display screen object may include a property to point to a previous display and another property to point to a next display screen. Each property, e.g.

the next display pointer on a display screen object, may be changed dynamically by using methods associated with the display screen object. One display screen object may serve as the starting point for processing reinsurance transaction. In one embodiment, the starting point for processing a reinsurance transaction may include acquiring a reinsurance claim number or a reinsurance policy identification number from an RC system user.

In one embodiment, during the processing of a reinsurance claim, a business rule and/or an RC system user input may determine that the reinsurance claim processing needs the execution of additional steps or tasks to continue the processing of the claim. The RC system user may provide inputs to the reinsurance transaction-processing program at any display screen 154 associated with a step included in the initial screen. The reinsurance claim processing software may dynamically modify the number of steps and/or the sequence of their execution to complete the claim processing transaction. An RC system user working at a client system may then iterate through the claim processing steps and arrive at an estimated value for the reinsurance claim.

In one embodiment, upon startup, the program may provide a graphical user interface (GUI) to display transaction processing related information on display screen 154. It may collect user inputs, entered by using user input devices 156 or 158, and associated with a reinsurance transaction. It may process the user inputs, access a reinsurance database 40, use the contents of the reinsurance database to estimate the reinsurance claim or premium, and store it in memory and/or reinsurance database. The program may display a value of the estimated reinsurance claim or premium on display screen 154. A user may view the display of the estimated reinsurance claim or premium on display screen 154, and may interactively make modifications, additions, and deletions to the estimated reinsurance claim or premium.

Computer system 150 may also include one or more user input devices, such as a keyboard 156, for entering data and commands into the reinsurance transaction-processing program. It may also include one or more cursor control devices 158 such as a

mouse for using a cursor to modify a reinsurance claim viewed on display screen 154. In response to the updating of the estimated reinsurance claim, the reinsurance transaction-processing program may store the updated reinsurance claim in the reinsurance database.

5 In one embodiment, the rule definitions, object properties, transaction values, etc. may be constructed and stored during the configuration of the RC system 150. Configuration of the RC system 150 may occur at the time the RC system software program(s) and database(s) are initially installed and set up for processing RC transactions. Configuration of the RC system 150 may also occur after the initial
10 configuration, which may be performed during the installation of the RC system. A configuration of the RC system that occurs after the initial configuration may be called a reconfiguration of the RC system. During reconfiguration, the rule definitions, object properties, transaction values, etc. defined or configured during the initial configuration may be modified or deleted, and new rule definitions, object properties, transaction
15 values, etc. may be added to the RC system 10. Some data may be processed differently in the RC system after a reconfiguration of the RC system than before the reconfiguration. For example, a business rule may be defined using a different rule structure after a reconfiguration. Thus the resulting algorithm computation returned to a calling program may be different than before the reconfiguration.

20 In one embodiment, the reinsurance transaction processing system may provide context-sensitive help for the processing steps. In one embodiment, the context-sensitive help for the step may be automatically invoked and displayed on display screen 154 when entering the step. In one embodiment, the user may interactively invoke context-sensitive
25 help for the step by selecting one or more interface items on the display screen 154 with a cursor control device 158 such as a mouse. In one embodiment, the user may interactively invoke context-sensitive help for the step by using an input device 156. For example, the user may select one or more keys or a combination of keys on a keyboard to activate context-sensitive help. The context-sensitive help for each processing step may

be unique, although content may appear in the context-sensitive help for two or more processing steps.

5 In one embodiment, information for the context sensitive help may be accessed from help database, included in database. Help database may include one or more one or more documents including information that may be useful to a user in performing the various processing steps associated with reinsurance transaction processing. Help database may also include one or more tables that provide access to the information in the documents. Each table may include a plurality of records or entries that may be used to
10 locate help information about processing steps and/or the elements in processing steps in the one or more documents in the help database.

In one embodiment, a search interface may be provided in the reinsurance transaction processing system 150. A user may enter in the search interface one or more
15 terms to be searched for in help database, included in database for the reinsurance transaction processing system 150. The user may then initiate the search for the one or more terms. The reinsurance transaction processing system may then search the help database for entries including at least one of the one or more terms. The reinsurance transaction processing system may locate one or more entries in the help database that
20 include at least one of the one or more terms. The reinsurance transaction processing system may then display information on display screen 154 from the located help database entries.

Figure 2: A Distributed Computing Environment

25

Figure 2 illustrates a distributed or enterprise computing environment according to one embodiment. A distributed computer system or enterprise 100 may include a plurality of computer systems, which are interconnected through one or more networks. Although one particular embodiment is shown in Figure 2, the distributed computer system 100 may

include a variety of heterogeneous computer systems and networks which are interconnected in a variety of ways and which run a variety of software applications and/or operating system software.

5 One or more local area networks (LANs) 104 may be included in the enterprise 100. A LAN 104 is a network that spans a relatively small area. Typically, a LAN 104 is confined to a single building or group of buildings. Each node (i.e., individual computer system or device) on a LAN 104 preferably has its own CPU with which it executes programs, and each node is also able to access data and devices anywhere on the LAN
10 104. The LAN 104 thus allows many users to share devices (e.g., printers) as well as data stored on file servers. The LAN 104 may be characterized by any of a variety of types of topology (i.e., the geometric arrangement of devices on the network), of protocols (i.e., the rules and encoding specifications for sending data, and whether the network uses a peer-to-peer or client/server architecture), and of media (e.g., twisted-pair
15 wire, coaxial cables, fiber optic cables, radio waves). As illustrated in Figure 2, the distributed computer system 100 may include one LAN 104. However, in alternate configurations the distributed computer system 100 may include a plurality of LANs 104 which are coupled to one another through a wide area network (WAN) 102. A WAN 102 is a network that spans a relatively large geographical area.

20 Each LAN 104 includes a plurality of interconnected computer systems and optionally one or more other devices: for example, one or more workstations 110a, one or more personal computers 112a, one or more laptop or notebook computer systems 114, one or more server computer systems 116, and one or more network printers 118. As
25 illustrated in Figure 2, an example LAN 104 may include one of each of computer systems 110a, 112a, 114, and 116, and one printer 118. The LAN 104 may be coupled to other computer systems and/or other devices and/or other LANs 104 through a WAN 102.

One or more mainframe computer systems 120 may be coupled to the distributed computer system 100. As shown in Figure 2, the mainframe 120 may be coupled to the distributed computer system 100 through the WAN 102, but alternatively one or more mainframes 120 may be coupled to the distributed computer system 100 through one or more LANs 104. As shown, the mainframe 120 may be coupled to a storage device or file server 124 and mainframe terminals 122a, 122b, and 122c. The mainframe terminals 122a, 122b, and 122c may access data stored in the storage device or file server 124 coupled to or included in the mainframe computer system 120.

The distributed computer system 100 may also include one or more computer systems, which are connected to the distributed computer system 100 through the WAN 102: as illustrated, a workstation 110b and a personal computer 112b. In other words, the enterprise 100 may optionally include one or more computer systems, which are not coupled to the distributed computer system 100 through a LAN 104. For example, the distributed computer system 100 may include computer systems, which are geographically remote and connected to the distributed computer system 100 through the Internet.

In one embodiment, the enterprise or distributed computer system 100 includes one or more computer systems configured for reinsurance transaction processing and/or developing reinsurance administration systems. The one or more computer systems for reinsurance transaction processing systems may be typical computer systems 150 as illustrated in Figure 1.

Figure 3a is one embodiment of reinsurance transaction processing software using object-oriented technology

The reinsurance software program may, in one embodiment, be implemented by using an object-oriented technology. The object-oriented technology may include, but not

be limited to, an object-oriented analysis and design, an object-oriented language and an object-oriented repository or a database. In one embodiment, the reinsurance software program may be implemented by using a legacy programming language, such as C. The object-oriented programs may interact with legacy code programs, and vice versa, to
5 process reinsurance transactions. In one embodiment, an object-oriented program may generate a new software module written in legacy code such as C.

Java™, Smalltalk and C++ are a few examples of object-oriented programming languages. In an object-oriented programming language, data and related methods can be grouped together or encapsulated to form an entity known as an object. The object is the
10 fundamental building block of object-oriented programming. The data structures within an object may alternately be referred to as the object's state, its properties, its attributes, its fields, or its variables. In the Java™ Language, the data structures are normally referred to as the variables of the object. In the Smalltalk object-oriented programming language, the data structures may be normally referred to as the properties of the object.
15 The procedures that operate on the variables are referred to in Java™ and Smalltalk as the methods of the object. The list of methods, associated with the object, may also be referred to as the list of operations. Unlike traditional programming, in object-oriented programming the operations on the variables may only be performed via the methods of the object. The properties and methods of an object may all be referred to as the members
20 of the object. Examples of a few methods that may be used when associated with a reinsurance business object, may include but not be limited to: a method for creating (setting up) a new business (reinsurance policy), a method for renewing (new insurance period) a business, a method for registering a new claim against a business, a method for entering technical booking items (premium, claim, commission) against one or more
25 business(es), a method for registering a remittance (payment) against a business partner, etc, etc.

In object-oriented programming, the grouping together of the properties and methods within an object is referred to as encapsulation. When the properties relating to an object and the methods that might affect the object are encapsulated within the object,

other entities usually do not have direct access to these data and procedures. The other entities instead call on the object itself to invoke its own methods and thereby operate on its own data. The encapsulation of the members of the object thereby provides some protection for the data within the object and prevents unauthorized, unwanted, or unintended manipulation of the data. This is sometimes referred to as data hiding. (The concept of data hiding through encapsulation should be distinguished from the hiding of properties in Java™ variable declarations, as explained in more detail below.)

In the Java™ language, if a user wants to hide the data within an object, the variable that contains the data is made private. Private properties within an object may only be accessed by the methods of the object. Because it may, in some cases, be inconvenient or impractical to require manipulation of certain data through the methods of the associated object, some properties may be made public. These public properties are directly accessible to entities other than the object with which the properties are associated. Thus, in practice, the properties within objects normally include some that are hidden or inaccessible and some that are public.

In the Smalltalk language, properties are always private. However, one can achieve the effect of a public property by providing a so-called "getter" and "setter" method for this property. These methods then allow (indirect) access to the property from entities other than the object with which the property is associated.

All objects in an object-oriented programming system belong to a class, which can be thought of as a category of like objects, which describes the characteristics of those objects. Each object is created as an instance of the class by a program. The objects may therefore be said to have been instantiated from the class. The class sets out properties and methods for objects that belong to that class. The definition of the class does not itself create any objects. The class may define initial values for its properties, and the class normally defines the methods associated with the class (i.e., includes the program code which is executed when a method is invoked.) The class may thereby provide all of the program code that will be used by objects in the class, hence

maximizing re-use of code, which is shared by objects in the class. Figure 3a represents one embodiment of an reinsurance object class and an instance of the object class, where the software to create object classes and instances of an object class are included in the reinsurance transaction processing software 30. Illustrated in Figure 3a, is one embodiment of a customer object class 300 and the creation of an instance 350 of the customer object class i.e., a customer object 300a. The customer class 300 may include a class identifier or a name 320, such as customer 321. The customer class 300 may also include one or more properties 330 such as name 331, date of birth 332, sex 333, address 334, phone number 335, occupation 336, policy number 337, etc. The customer class 300 may also include one or more methods 340 associated with the class 300 such as calculate_age 341, rating_factor 342, accident_history 343, etc. For example, the calculate_age method 341, when invoked, may calculate the current age of a customer. The method may return an argument containing the applicant's age. Creating an instance 350 of the customer class 300 may create a customer object 300a. Specific properties 330a associated with the instantiated object 300a, such as object ID = John_Doe 321a, name = John Doe 331a, sex = M 333a, etc. may be configured. Additional examples of object classes may include a business class, a reinsurance contract class, a reinsurance policy class, a reinsurance claim class, etc. As another example, a business rule may be defined as a class.

In most object-oriented languages, classes may be hierarchical. That is, some classes may be subclasses of a higher class, also known as a superclass. In the Java™ language, the subclass is said to extend the superclass. Alternatively, the superclass is said to be extended by the subclass. For the purposes of this disclosure, a subclass is considered to extend all or any of the classes that are above it in the hierarchy. It may also be said that the subclass directly extends the class immediately above it in the hierarchy, and indirectly extends higher classes in the hierarchy. For example, if a parent class is extended by a first subclass and that subclass is in turn extended by a second subclass, the second subclass can be said to extend the parent class as well as the first subclass.

Problem solving may be accomplished by sending message(s) to objects. A message may name a method and may optionally include other arguments associated with the method. When a message is sent to an object, the method is looked up in the object's class to find out how to perform that operation on the given object. If the method is not defined for the object's class, it may be looked for in its superclass and so on up the class hierarchy until it is found or there is no higher superclass. Message sends may return a result object, which may be an error, as in the case where no superclass defines the requested method.

The hierarchical definition of classes and subclasses based on shared properties and methods is very useful. A subclass includes all the properties and methods in the class of which it is a member (its parent class). The subclass is said to inherit the properties and methods of its parent class. This property is useful in defining subclasses because only those properties and methods that do not appear in the parent class need to be defined in the subclass (although properties or methods which appear in the parent class may be redefined in the subclass.) This allows the code written in the parent classes to be re-used so that the programmer does not have to rewrite or cut and paste code into each new subclass. Methods that are defined in the parent class may, however, be redefined in subclasses. This is referred to as overriding or hiding the previously defined method(s). In Smalltalk, for example, when a class A has a subclass B which overrides a method x defined in A, then, when the message x is sent to an instance of class B, class B's implementation of method x will be used. In some object-oriented languages, by redefining a variable that has already been defined in a superclass, the programmer may hide the previously defined variable (which is distinct from the object-oriented data hiding concept inherent in encapsulation.) In some object-oriented languages, subclasses may inherit properties and methods from several classes. This is called multiple inheritance. If a subclass can only inherit from one parent class, this is called single inheritance. Inheritance may also be dynamic and/or shared. Specifically, dynamic inheritance refers to the ability to add, delete, or change parents from objects (or classes) at run-time. Multiple inheritance brings up the possibility for a class to appear as a parent

more than once in a class graph (shared or repeated inheritance), and there is then a potential to share that class. The Java™ and Smalltalk languages are characterized by single inheritance, not multiple.

Hierarchical class structure also allows the programmer to take advantage of a property referred to as polymorphism. Polymorphism is a mechanism by which various objects may be handled in the same way externally, even though there are differences in the way they are handled internally. In other words, the interface that the different objects present to an external entity is the same for each object, but the details of each object's implementation may vary. This allows objects instantiated from different subclasses to be handled identically even though the subclasses are not identical. For example, assume that a drawing program implements a class for shapes, a subclass for circles, and a subclass for squares, each of which has a method called draw(). While draw() will be implemented differently for the circle subclass and the square subclass, the drawing program does not have to know the details of how a shape will be drawn, or even which of the shapes is to be drawn. The drawing program simply calls the draw() method for the object to be drawn and the implementation defined in the object's class will be used.

Figure 3b: Layered Architecture of the Reinsurance Administration System

Figure 3b illustrates a layered architecture of a reinsurance administration system according to one embodiment. The reinsurance administration system may include multiple layers or levels. As shown in Figure 3b, the reinsurance administration system may include five layers: an interface layer 302, a business process layer 322, a domain layer 342, a persistency layer 362, and a relational database 382. Each layer may include objects that collaborate in solving the specific responsibilities of the layer. As used herein, an “object” is a unit of functionality that may include both data and methods for operating on the data.

In one embodiment, the interface layer 302 may include objects, which are responsible for presenting data to the user or to another system. The objects of the interface layer 302 may be responsible for providing input to the business process layer 322. For example, the interface layer 302 may include objects such as one or more graphical user interface (GUI) objects 304, one or more world wide web (WWW) objects 306, and one or more electronic data interchange (EDI) objects 308.

In one embodiment, the business process layer 322 may include objects, which are responsible for executing the steps of a logical unit of work. For example, a logical unit of work may include executing validation, committing a transaction, and rolling back a transaction. The business process layer 322 may include objects such as one or more business process objects 324, one or more security objects 326, and one or more validation objects 328. A business process object may be responsible for user access privileges or other security issues, obtaining or generating domain data, validation of domain data, and adding, modifying, and deleting data held by a database. In one embodiment, business processes are coordinators and do not generally implement the actual code, which performs basic tasks. The actual code may be implemented in domain objects 344 or in support processes, which are created and used by a business process.

In one embodiment, the domain layer 342 may include objects, which are responsible for holding application data and implementing basic business functionality. For example, the domain layer 342 may include objects such as domain objects 344 and helper objects 346. The domain objects 344 and helper objects 346 may collaborate to provide a concrete representation or definition of the business objects 324. The objects in the domain layer may implement reinsurance functionality without reference to implementation details such as persistency and user interfaces.

In one embodiment, the persistency layer 362 may include objects, which are

responsible for enabling communication between the domain layer 342 and the relational database 382. For example, the persistency layer 362 may convert domain objects to rows in the relational tables, may provide an object caching mechanism, and may manage memory transactions. The persistency layer 362 may include objects such as one or more objects for transaction handling 364, one or more objects for database access 366, one or more objects for object caching 368, and one or more objects for database mapping 370.

In one embodiment, the relational database 382 may include one or more database tables 384. The relational database may communicate with the persistency layer 362 through the Open Database Connectivity (ODBC) standard for database access. In one embodiment, the database 382 may be any database which is ODBC-compliant. The database tables 384 may store objects from the other layers in a persistent form.

Figure 4: Development of a Reinsurance Administration System

Figure 4 is an illustration of a development process for developing a reinsurance administration system according to one embodiment. The reinsurance administration system may be developed with software tools such as compilers, object-oriented development environments, and other suitable tools. In one embodiment, a Smalltalk development environment may be used. The development process may re-use program code for increased efficiency, reduced development time, and reduced cost.

In step 402, functional requirements may be defined. The results of step 402 may include at least one use case 404 and a data dictionary 406. As used herein, a “use case” may define a particular business function or grouping of business functions. In other words, use cases may correspond to business requirements for the reinsurance administration system. In one embodiment, the reinsurance administration system may include use cases for “basic” or “support” data such as a reference data administration use case, a system parameter administration use case, a security administration use case, a

currency administration use case, a reporting unit administration use case, a risk zone administration use case, and a location administration use case. Use cases may also assist a user of the reinsurance administration system in carrying out his or her daily work. For example, these use cases may include a use case for creating or setting up a new business (e.g., a reinsurance policy), a use case for renewing a business (e.g., a new insurance period), a use case for registering a new claim against a business, a use case for entering technical booking items (e.g., premium, claim, commission) against one or more business(es), and a use case for registering a remittance (i.e., payment) against a business partner.

10

In step 412, prototyping and GUI design may be performed. The results of step 412 may include graphical user interfaces (GUIs) 414 and at least one prototype 416. A prototype may be a demonstration program that may not include the complete functionality of the final version of the program, but rather may demonstrate the capabilities planned for the final version.

15

In step 422, modeling may take place. Step 422 may produce a domain model 424 and a business process model 426. The domain model 424 and/or business process model 426 may embody one or more use cases as developed in step 402.

20

In step 432, application parts may be implemented and tested. The results of step 432 may include at least one new application part 434. An application part may be one or more new parts 434 or a combination of such parts. The application part 434 may be placed in a parts inventory 436 with other application parts. The application parts in the parts inventory may be used and re-used to build one or more application programs. In one embodiment, the system for developing a reinsurance administration system may include a variety of design patterns and reusable parts in the parts inventory 436. For example, the system for developing a reinsurance administration system may include a general pattern for building "Find..." components (e.g., Find Business, Find Business

25

Partner, Find Claims, etc.), domain class mediators (for retrieving items such as currencies, countries, states, country groups, etc.), and/or a general pattern for integrating new business condition components within the business structure. The system for developing a reinsurance administration system may also include reusable GUI parts such as a calendar widget for entering dates, a select/available widget, domain object drop-down lists, and many larger-grain GUI parts that may include multiple basic widgets.

In step 442, the parts may be assembled to build a computer-executable reinsurance administration system 444. Steps 402, 412, 422, 432, and 442 may be repeated a plurality of times to produce a plurality of reinsurance administration systems 444. The plurality of reinsurance administration systems may include multiple versions of one reinsurance administration system 444. The design process illustrated in Figure 4 may therefore include the ability to extend and customize the reinsurance administration system to accommodate new and specialized requirements.

Using Frameworks to Develop a Reinsurance Administration System

One or more frameworks may be utilized by a developer to build one or more reinsurance management systems. The frameworks may include functionality for solving general and recurring problems. Use of the frameworks during development may therefore decrease the time and expense associated with product development.

A framework may include a collection of classes, which contain logic for solving problems that are common to a range of functional requirements. To solve a specific requirement which falls within the range of requirements for which a framework was designed, a developer may create subclasses of the classes contained within a framework. While the inherited properties of the classes will provide the solutions to the general aspects of the problems to be solved, the developer may implement solutions to special requirements by providing specialized code and data. The developer may add additional

behavior by adding methods to the subclasses, and/or by making the classes collaborate with other, specialized classes. A framework may provide general methods, called “hook” methods, that may be overridden by the developer to identify what action the system should take at a certain point. Thus, a framework may provide patterns and templates for how to solve common requirements. By doing so, a framework may permit a group of developers to avoid solving the same class of problem in different ways.

For example, for most windows in a graphical user interface, a common requirement is that it must be possible to close the windows. Therefore, a common feature of most windows is a “close” button or another means for the user to close the windows. An additional requirement may be that the system should ask the user whether data changes should be saved when the user chooses to close a window. These requirements may be solved by reusable portions of a framework to avoid re-implementing the requirement for each new window. The framework may also provide a hook method that may be overridden by a developer to implement saving data changes upon the closing of a window.

The frameworks in the system for developing reinsurance administration systems may include technical frameworks and industry frameworks. Generally speaking, technical frameworks may implement common, general business behavior. Technical frameworks may be reusable in a variety of types of software program applications such as reinsurance administration programs. Technical frameworks may enable developers to focus on implementing business rules. Technical frameworks may enable inexperienced developers to become productive in record time.

Industry frameworks may implement common behavior and object models for a particular business area. An object model framework may abstract and encapsulate complex business structures. In one embodiment, the object model framework may capture the multiple dimensions of a contract and hide the multi-dimensional aspect of a

contract from subsequent domain modeling. The multiple dimensions of a contract may include multiple reinsurance periods, multiple life cycle phases per insured period, multiple sections per life cycle phase, multiple amendments per life cycle phase, and contract conditions per section and per amendment, wherein the sections may be organized in a hierarchy. In one embodiment, a behavior framework may abstract and encapsulate business rules for managing the multiple contract dimensions into an abstract framework. A behavior framework may provide inherited functional support to core business components. A behavior framework may include support for component-based development.

10

The frameworks may include a business process framework. As used herein, a “business process framework” may include a framework for developing business processes. As used herein, a “business process” may include a computer-based representation of a use case or business function to be performed in a line of business.

15 The business process framework may allow developers to focus their attention on business-specific requirements, while the framework handles common, “technical” requirements. In short, the business process framework may allow a developer to quickly assemble a business process component. The business process framework may include a plurality of high-level business process classes. The high-level business process classes

20 included in the business process framework may include default implementations for hook methods which subclasses may override. In one embodiment, a developer may create a new business process component by subclassing another business process class from an existing business process class. The many classes that together make up a reinsurance software system may be arranged in a hierarchy. At the top of the hierarchy

25 may sits the most general, abstract business process class. This class may be subclassed by one or more other classes, some of which may be abstract. Classes which are not abstract are considered to be concrete. This process of subclassing may go on virtually indefinitely. The hierarchy of (sub)classes need not be balanced; that is, concrete classes may appear at various levels. A class which has multiple subclasses may serve to capture

behavior (i.e., logic) that is common to its subclasses. Unlike the concrete classes, the abstract classes are not intended to be instantiated as objects at runtime.

5 A business process class may implement its required behavior by identifying appropriate support processes for its configuration and by overriding selected hook instance methods provided by the framework (i.e., in one embodiment, the topmost business process classes in the hierarchy). As used herein, “support processes” may include computer-based processes for performing tasks associated with a business process. As used herein, “hook methods” may include methods in a parent class which
10 provide default methods to be inherited by a child class and which may be overridden by the child class. The business process framework may provide many support process methods, classes, and subclasses which may implement behavior for: precondition checks (for data requirements to start a business process), security checks (for user access to functions and data), data entry, graphical user interfaces (GUIs), data and syntax
15 validation, persistent data storage, committing changes to a database, rolling back changes to a database, and other suitable functions.

As used herein, “support processes for data validation” may include support processes for validating data prior to committing the data to a database. As used herein,
20 “support processes for security” may include support processes for controlling the access of a user to particular business functions. As used herein, “support processes for persistent data storage” may include support processes for storing data in a database or other data storage system. As used herein, “support processes for precondition checks” may include support processes for determining whether a business process has the
25 necessary initial data to begin its execution. As used herein, “support processes for data entry” may include support processes for obtaining data from a user through a user interface, from another computer-based process on the same computer system, and/or from another computer system over a network.

In one embodiment, the business process framework may automatically handle requirements such as, for example: opening a specified window, adding Cancel/OK/Save/Help buttons to an open window, providing the behavior for buttons added to a window, stepping through the different steps or stages of the business process template (e.g., pre-execution, data entry, validation, pre-commit, commit, post-commit) and executing the specific logic that may be specified by a developer for each phase, performing a rollback of objects in memory upon error or cancellation by a user, performing validation, committing changes to the database, performing write-lock checking, and/or logging and displaying error messages.

10

Figure 5: Using a Business Process Framework

Figure 5 is a flowchart illustrating the use of a business process framework by developers to build application programs for reinsurance administration with increased re-use of existing functionality and decreased development time. In various embodiments, the steps shown in Figure 5 may be performed in a different order. In one embodiment, the method may include deriving or subclassing the plurality of business process classes from the business process framework in step 502. One or more business process subclasses may be derived from one or more abstract business process classes, wherein the business process framework may include the one or more abstract business process classes. In one embodiment, business process subclasses may be derived from concrete business process classes as well as abstract business process classes.

The business process framework may include a plurality of support processes. In step 504, one or more of the support processes may be specified for each of the business process subclasses. The support processes may include support processes for data validation, support processes for performing syntax validation, support processes for graphical user interface elements, support processes for persistent data storage, support processes for performing precondition checks, and/or support processes for other suitable

functions. In one embodiment, identifying one or more of the support processes for each of the business process subclasses may include overriding one or more of the hook methods for each of the business process subclasses.

5 The business process framework may include a plurality of hook methods. The hook methods may permit the developer to access various points during the execution of the business process. In step 506, the developer may override one or more hook methods for a business process subclass. By overriding the hook methods, the developer may identify methods to be performed at various points during execution of the business
10 process. Overriding one or more of the hook methods for each of the business process subclasses may include replacing one or more of the hook methods derived from the abstract business process classes with one or more new methods for each of the business process subclasses. The hook methods may include, for example, methods to be executed during initialization of the business process, methods to be executed prior to execution of
15 a data entry support process, methods to be executed during execution of data entry without a user interface, methods to be executed prior to execution of a database commit support process, and/or methods to be executed prior to execution of a database rollback support process. As will be apparent to one skilled in the art, many more hook methods may be available to be overridden by a business process subclass.

20 The common functionality of the business process framework may include an order for executing steps in a business process. The steps in the business process may include pre-execution, data entry, data validation, pre-commit to a database, commit to the database, and post-commit to the database. These steps are further discussed with
25 reference to Figures 6a and 6b. The common functionality of the business process framework may include behavior for buttons in a graphical user interface, formatting and displaying windows in a graphical user interface, methods or variables of the abstract business process classes which are inherited by the business process subclasses, logging and displaying error messages, committing changes to a database, and/or rolling back

changes to a database. In one embodiment, a first business process subclass may be configured to invoke a second business process subclass.

In step 510, the business process subclasses may be combined to build one or more application programs, such as application programs for reinsurance administration. The resulting application program(s) may be stored on a storage device. When the resulting application program(s) are executed, business process objects will be instantiated from the business process subclasses created by the developer.

10 **Figures 6a and 6b: The Life of a Typical Business Process**

Figures 6a and 6b are state diagrams illustrating the life of a typical business process according to one embodiment. In one embodiment, the business process framework may implement a business process state transition machine and control the sequencing of steps in a business process. A developer may override hook methods to identify methods to be performed at various stages in the sequence of steps in life of the business process.

In one embodiment, when a new business process is created in step 602 by instantiating a business process object from the corresponding class, it will execute the states shown in Figure 6a and stop in the ready state 612. These states or steps may include initialization 604, installation of a description 606, installation of a configuration 608, and transferal of settings from the description 610. The business process may remain indefinitely in this state until it receives an execute command. Once execution starts, there are many paths, which the process can follow, and a common path is illustrated in Figure 6b. If the business process uses an interactive GUI interface for the data entry, then the business process may proceed from the ready state 612 through to the execute data entry state 632 and remain there indefinitely until the user chooses to save, cancel, or close. The other steps or states may include initial execution 622, getting an

edit token if required 624, performing a preconditions check 626, performing a security check 628, and transferring arguments from a parent process 632. Once the user chooses to exit the data entry state 632, the business process may execute either a validate 633 and commit 634 or a rollback 636 and then do the finishing actions 638.

5

In one embodiment, performing the security check 628 may include determining whether a user has the proper access rights to access a particular use case. Support processes for access security may guarantee no access security, access security on a business process level, or access security with domain restriction. A support process for access security on a business process level may determine whether the user has the access right to execute a business process, wherein the business processes in the reinsurance administration system are each linked to a particular access right. A support process for access security with domain restriction may be similar to a support process for access security on a business process level but may also test domain restrictions for the user.

15

As discussed with reference to Figure 5, one or more hook methods may be provided to allow the developer to access clean points in the execution of the business process. A hookInitialize method 652 may be executed during initialization of the new business process. This method may be sent before the description is available and before any support processes are created. A hookPreExecuteDataEntry method 654 may be executed just prior to the execution of the data entry support process. In one embodiment, if the business process has an associated window, the window will not be opened until the data entry support process is executed. In one embodiment, invocation of a child business process from this hook method 654 is not allowed.

25

In one embodiment, a hookDelegatedExecuteNoDataEntry method 656 may only be used if the business process is configured with a CnuNoDataEntry support process and will occur during the execute data entry phase 632. This hook 656 is provided to allow a business process which operates without a user interface to perform some required

processing before the business process commits. In one embodiment, invocation of a child business process from this hook method 656 is allowed.

In one embodiment, a hookPreExecuteCommit method 658 may be executed prior to the execution of the commit support process 634. In one embodiment, invocation of a child business process from this hook method 658 is allowed. A hookPreExecuteRollback method 660 may be executed just prior to the execution of the rollback support process 636. In one embodiment, invocation of a child business process from this hook method 660 is allowed.

In one embodiment, a plurality of hooks may be used by the developer to modify the complex process of validation 633. Validation may include syntax validation of the business process, a domain restriction security test, and any additional, business-process-specific validation that the developer may have specified for the business process. Syntax validation of the business process may itself include syntax validation of those domain objects the business process has specified in one of its hooks, such as hookCompleteSyntaxDescriptor 662. The domain restriction security test may use hooks such as hookDomainObjectsToSecurityCheckDescriptor 664 to check that the data values entered into the domain object(s) by the user are in accordance with the privileges stored in the user's logon profile for the reinsurance administration system. The additional, business-process-specific validation may use hooks such as hookValidationProcessClass 666.

Standard Methods to Override and Standard Classes to Subclass

The business process framework may include a plurality of standard methods that may be overridden and a plurality of standard classes that may be subclassed by developers. Several examples are discussed as follows; as will be apparent to one skilled in the art, many more hook methods and classes may be used. A hookCommitProcessClass may provide the functionality to handle the commit and

transaction allocation for a business process. A hookPreExecuteDataEntry method may be used to set up anything that is needed before the data processing process is executed. A business process that is responsible for the creation of a new object would normally override the hookPreExecuteDataEntry method in order to implement the instantiation of the new object. As discussed with reference to Figure 6b, a hookValidationProcessClass, hookDomainObjectsToSecurityCheckDescriptor, and hookCompleteSyntaxDescriptor may provide the functionality for doing any necessary validation before any data is to be committed. In one embodiment, data will be committed only once all validation executes successfully.

10

A hookGuiInterfaceSpecificationInSession may provide the specification of what the GUI interface is to look like. From this specification, a window may be built depending on how the business process is used. A guiInterfaceWindowLabel may return a string label to use to place on the window that may be scheduled for the business process. A hookGuiInterfaceClass may hold all of the functionality that is GUI-specific to how a business process is handled. Functions such as sorting lists and converting a radio button into a domain value may be handled by an instance of this class. In one embodiment, for example, any dynamic label of a window label would be handled here.

15

20 **Figure 7a: Illustrates One Embodiment of an Inheritance View of a Hierarchy of Business Objects and Figure 7b: Illustrates One Embodiment of an Inheritance View of a Section Object**

A typical reinsurance contract may include, but not be limited to, several attributes such as an insured period, i.e., a period representing the term of the reinsurance contract, contract conditions, insurance coverage, partners, amendments, premiums, etc. Each attribute may vary independently of the other attributes. Thus a reinsurance contract may be viewed to have multiple dimensions depending on the number of independent variable attributes included in the reinsurance contract. For example, the multiple

25

dimensions of a contract may include multiple reinsurance periods, multiple life cycle phases per insured period, multiple amendments per life cycle phase, etc.

5 In one embodiment, the reinsurance transaction processing software may include an object-oriented framework, the multi-dimensional reinsurance contract framework. In one embodiment, the multi-dimensional reinsurance contract framework may permit the handling of the four dimensions of a reinsurance contract. For example, the first dimension may handle different versions of the reinsurance contract (e.g., yearly periods). The second dimension may handle multi-level specialization hierarchy (e.g.,
10 different conditions for different sections of the reinsurance contract). The third dimension may handle the change of conditions within a period, potentially only for one section and the fourth dimension may process inheritance of contract structure and conditions vertically from a parent section to a child section and/or horizontally from a section of one reinsurance contract to the same section in another, linked reinsurance
15 contract. This framework may use an object model to capture the four dimensions and hide the multi-dimensionality from subsequent domain modeling and the design of additional reinsurance contract components. This framework may also encapsulate business rules for managing the four reinsurance contract dimensions into an abstract framework.

20 In one embodiment, the reinsurance transaction processing software may include an object-oriented framework to process one or more reinsurance business transactions. The object-oriented framework 5010 may include a collection of classes of objects. In one embodiment, an entire reinsurance business transaction or a part thereof may be
25 represented as a contract class. A contract object 536 may be created by instantiating a contract class. In one embodiment, a contract class may be a concrete subclass of a hierarchy of contract classes. The concrete subclass may capture special properties and methods for a specific level of business. In one embodiment, a user may configure customized methods and properties associated with the specific level of business.

Examples of the level of business may include assumed business, outward cedent's contract, own retention, etc. The reinsurance contract object 536 may represent one reinsurance contract, including all terms and conditions, all history and all attributes associated with the contract. The reinsurance contract object 536 may include properties and methods associated with the reinsurance contract. Examples of properties associated with the reinsurance contract object 536 may include, but not limited to, reinsurance period, life cycle phase per insured period, section(s) per life cycle phase, amendment(s) per life cycle phase, contract conditions per section and per amendment.

In one embodiment, an insured period object 550 may be created by instantiating an insured period class. In one embodiment, an insured period class may be a concrete subclass of a hierarchy of insured period classes. The concrete subclass may capture special properties and methods for a specific level of business. If the insured period is yearly then it may be also referred to as the underwriting year. In one embodiment, the insured period object 550 may own all terms and conditions, all history and all of its child objects for one calendar period of the reinsurance contract.

In one embodiment, a life cycle phase object 552 may be created by instantiating a life cycle phase class. In one embodiment, a life cycle phase class may be a concrete subclass of a hierarchy of life cycle phase classes. Typical life cycle phases included in a reinsurance contract may include new agreement offered, offer accepted, offer declined, new quote requested, renewal offered, etc. In one embodiment, the life cycle phase object 552 may own all terms and conditions, all history and all child objects for one phase of one calendar period of the reinsurance contract.

In one embodiment, a section object 554 may be created by instantiating a section class. In one embodiment, a section class may be a concrete subclass of a hierarchy of section classes. Sections may be organized in a hierarchy. The highest-level section object 554 may also be referred to as the main or root section 701a in Figure 7b. The root

section object 554 may have zero or more child section 702a objects. Each child section object 702a may have zero or more sub-child section objects. Each section object 554 may have a classification 703a with one or more categorization values. Typical categories 704a are main class of business, class of business, risk area country, type of participation, etc. Typical categorization values 705a are casualty, workmen's compensation, aviation, marine, marine hull, marine machinery, etc. The classification of a child section 702a must be a subset of the classification of its parent section. In one embodiment, the section object 554 may own all terms and conditions, all history and all child objects for one section of one phase of one calendar period of the reinsurance contract.

10

In one embodiment, an amendment object 560 may be created by instantiating an amendment class. In one embodiment, an amendment class may be a concrete subclass of a hierarchy of amendment classes. In one embodiment, the amendment class may include an abstract class and two concrete subclasses of the abstract class. The first amendment concrete subclass may be used to define the initial terms and conditions. The second amendment concrete subclass may be used to amend one or more conditions within the insured period. Amendments may be shared amongst all life cycle phases within an insured period. In one embodiment, when a condition, e.g., a premium condition, is amended, the insured period gets divided into a set of non-overlapping effective periods that together make up the insured period. In one embodiment, conditions such as premium, limit, deduction, share, etc. may be amended. Section classification and protection may also be amended. Examples of protection, which may be amended, may include treaty, faculty, proportional/non-proportional protection assignments, retrocession contracts, own retention, etc.

25

In one embodiment, the reinsurance transaction processing software may include an object-oriented framework, the condition component framework. This framework, i.e., the condition component framework, may permit the addition and/or modification of condition components of the reinsurance contract. These components may comprise, for

example, premium limits, consolidation conditions, and other conditions for reinsurance contracts. In one embodiment, the components may be implemented as reusable objects.

In one embodiment, the reinsurance framework 5010 may include one or more classes that represent contract properties that may be shared. An inheritable object may be created by instantiating an inheritable class. Each inheritable object may have a parent or an owning section. In addition, each inheritable object may be connected to zero or more other sections in inherited or shared mode. Inheritance may be vertical or horizontal. Vertical inheritance means inheriting from a parent section. Horizontal inheritance means inheriting from the same section in another, linked reinsurance contract. An inheritable object may be read/viewed both from the context of its owning section and its inheritor sections. An inheritable object may only be edited in the context of its owning section. Examples of shared inheritable objects include a condition object 572, a protection object 592, and a section classification object 582.

Figures 8a through 8d – Illustrates One Embodiment of a Graphical User Interface for Processing a Reinsurance Business Transaction

A typical reinsurance business transaction may include, but not be limited to, submitting a quotation for a reinsurance policy, calculating a premium associated with a reinsurance policy, issuing a contract for a reinsurance policy, processing a claim, etc. A reinsurance contract may be one embodiment of a reinsurance transaction. A typical reinsurance contract may include, but not be limited to, several attributes such as an insured period, i.e., a period representing the term of the reinsurance contract, contract conditions, insurance coverage, partners, amendments, premiums, etc. Each attribute may vary independently of the other attributes. Thus a reinsurance contract may be viewed to have multiple dimensions depending on the number of independent variable attributes included in the reinsurance contract. For example, the multiple dimensions of a contract may include multiple reinsurance periods, multiple life cycle phases per insured period, multiple amendments per life cycle phase, etc.

5 In one embodiment, the reinsurance transaction processing software may include an object-oriented framework to process all reinsurance business transactions. In one embodiment, the graphical user interface software included in the reinsurance transaction processing software may use objects to manage various user interface functions. The object-oriented framework may include several classes of objects. In one embodiment, an entire reinsurance business transaction or a part thereof may be represented as a reinsurance contract object.

10 In one embodiment, the reinsurance transaction processing computer system may include a display screen 154, a user input device such as a keyboard 156 and a graphical user interface software to interact with the user. Figures 8a through 8d illustrate one embodiment of a display included in a graphical user interface to manage reinsurance business transactions. In one embodiment, the management of a reinsurance business transaction may include the management of a reinsurance contract. Examples of displays used to manage reinsurance contracts may include, but not be limited to, configuration displays to define initial terms and conditions, displays to modify or amend the contract, displays to compute premiums, time periods, etc. In one embodiment, a display may be defined as a display class and specific display objects may be created as instances of the display classes.

Display Layout

25 In one embodiment, a display or a window may include one or more window panels 804 and a navigational tool 802. In one embodiment, a window panel may occupy a portion of the display screen or an entire display screen. The window panel, which may be also referred to as a window or a panel, may be opened, closed, minimized, maximized, moved, resized, etc. The one or more windows, which may be arranged in a tiled or overlapping manner, may display data 806 associated with the reinsurance

contract. In one embodiment, the arrangement and/or size of each window may be defined by the user or may be defaulted to a fixed layout by the reinsurance system software. In one embodiment, each window may display data associated with the methods and/or the properties of an object handled by the reinsurance system software, including a reinsurance contract object. Some or all of the data displayed may be modifiable by the user. In one embodiment, modifiable data 806 may be displayed on a white background. In one embodiment, non-modifiable data may be displayed on a gray background.

10 In one embodiment, the navigational tool 802 may provide one or more user interface items 808 such as icons, buttons, pull down menus, etc. to enable the user in navigating through various displays. In one embodiment, all graphical user interface functions accomplished by using the user interface items may also be accomplished by using special function keys on a keyboard 156, by a top level pull-down menu, etc. The
15 navigational tool 802, in one embodiment, may include one or more tool panels 802a, 802b and 802c. In one embodiment, the position of the navigational tool may remain fixed regardless of the other windows included in the display. Each of the one or more tool panels 802a, 802b and 802c may provide navigational tools associated with displaying a particular reinsurance contract function. For example in one embodiment, a
20 tool panel 802c may include up/down and/or left/right arrow buttons 802cb and an associated display field 802ca to assist the user in navigating through a section hierarchy. On selecting up/down and/or left/right arrow 802cb, the value displayed in the associated display field 802ca may change according to the user selection. If during the navigation process, the user has already reached at the top of a hierarchy then the tool panel 802c,
25 which includes the up button 802cb, may be partially disabled or displayed in a lighter color.

In one embodiment, the interface item or items may be provided to the user to navigate within a display, for example, to select various ranges for a variable displayed

within a tool panel 802. Interface items may be items displayed graphically on the window panel 804 and/or the tool panel 802 portion of the screen (for example, as icons or buttons) and may be selectable using input/output devices such as a mouse 158, joystick, or arrow keys on a keyboard 156. Interface items 808 may also be keyboard selections such as function keys or key combinations. For example, a button may be provided that allows the user to scroll down the list of valid entries in a pull-down menu.

The navigational tool 802 may be configurable in one embodiment. Each of the interface items 808 such as a button with an icon may represent a particular graphical user interface function. For example, the user may click on a particular button or icon included in the navigational tool 802 to navigate to a particular display. In one embodiment, clicking of an interface item 808 may initiate an action such as performing a calculation, printing of a report, etc. The user may configure the particular tool button by linking the click action to executing a particular graphical user interface function.

In one embodiment, the graphical user interface may include a hierarchy of displays related to a reinsurance contract. In one embodiment, by entering a reinsurance contract number into the display screen the user may access an initial contract display. In one embodiment, the reinsurance software may automatically display a list of all available contracts as the initial display. The initial display may represent the highest-level display within the hierarchy of displays. The navigational tool 802 may display the user interface items 808 to enable the user to select any dimension of the contract. In one embodiment, the dimensions of the reinsurance contract may include, but not be limited to, an insured period 802b, a life cycle phase, a section 802c, an amendment, and inheritable conditions, section classification and protection. In one embodiment, each of the display included in the hierarchy of displays may describe a dimension of the reinsurance contract.

In one embodiment, a user may initiate a business transaction to modify a

downstream section, i.e., not the root or the main business section, to an existing reinsurance contract. On entering a valid contract number the initial display, e.g., a first window, may be displayed on the screen. The user may click on a down arrow 802cb from the section tool panel 802c included in the navigational tool 802, e.g. select a first interface item. In response, a second window may be displayed which may include data on the downstream or the child section, i.e., display downstream section data consistent with the user selected down arrow input. An authorized user may modify the modifiable data 806 displayed in the second window and on completing the modifications may click on a back arrow 802aa included in the tool panel 802a, e.g. selecting the second interface item, to return to the first window.

Various embodiments may further include receiving or storing instructions and/or data implemented in accordance with the foregoing description upon a carrier medium. Suitable carrier media include memory media or storage media such as magnetic or optical media, e.g., disk or CD-ROM, as well as signals or transmission media such as electrical, electromagnetic, or digital signals, conveyed via a communication medium such as networks and/or a wireless link.

Although the system and method of the present invention have been described in connection with several embodiments, the invention is not intended to be limited to the specific forms set forth herein, but on the contrary, it is intended to cover such alternatives, modifications, and equivalents as can be reasonably included within the spirit and scope of the invention as defined by the appended claims.